



Web Services

Dr. Malgorzata Mochol
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mochol@inf.fu-berlin.de

Vorlesungen – 7 Termine	Übungen – 5 Termine
XML-Grundlagen einsch. Namenräume	XML-Syntax, Namensräume
DTD & XML-Schema	DTD
XML-Schema im Detail	XML Schema
SAX & DOM Parser	
XPath	XPath & Co.
XSLT	
XLST & XLST 2.0	XSLT

Vorlesungs- -termine	Vorlesung (4 + 1 + 1)	Übung – 2 Termine	Übungs- termine
10.06. (heute)	Web Services, RPCs vs. Messaging		
17.06.	SOAP im Detail	SOAP	24./25.06
24.06.	WSDL im Detail	WSDL	
01.07.	Web Services in der Praxis & Ausblick		01./02.07
08.07.	Rückblick		
15.07.	Klausur		

heutige Vorlesung

- Was sind Web Services?
- Web Services – Basistechnologien:
 - SOAP
 - WSDL
 - UDDI
- Enterprise Application Integration (EAI)
- Dienstorientierte Architektur (SOA)
- RPC vs. Messaging



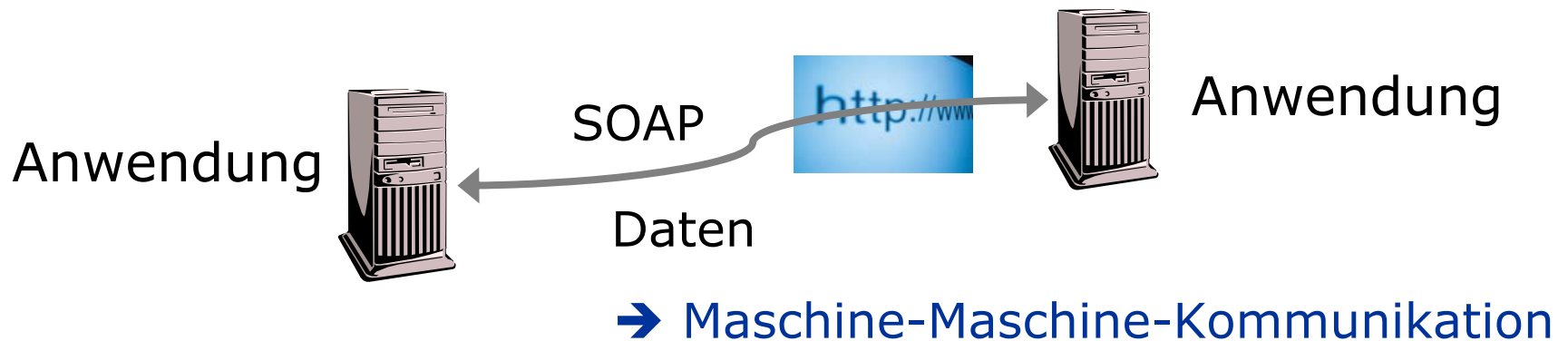
Was sind Web Services?

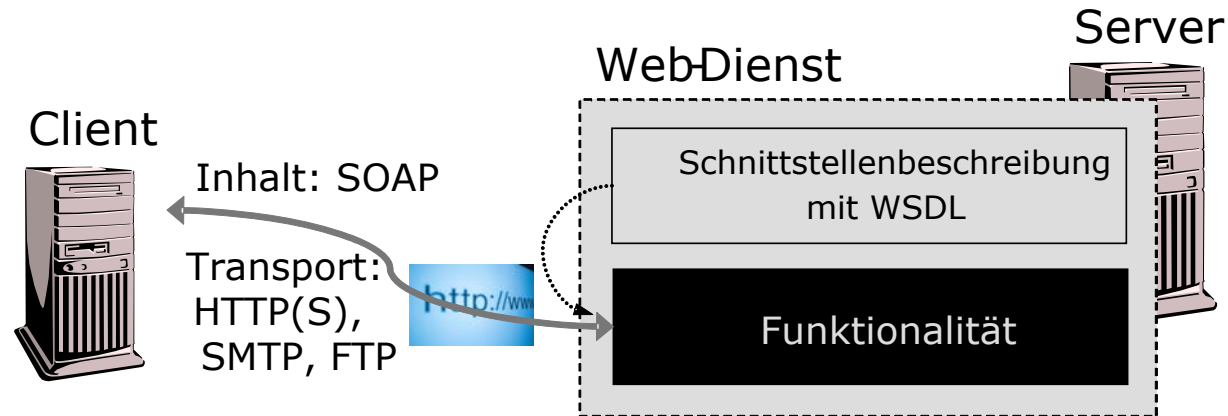
- **Email:** Mensch-Mensch-Kommunikation
 - Versendung von uninterpretiertem Text
- **WWW:** Mensch-Computer-Kommunikation
 - Mensch – aktiv
 - konsumiert & gibt an, was er als nächstes lesen möchte.
 - Computer – passiv
- **Web Service:** Computer-Computer-Kommunikation
 - Computer erbringt für einen anderen eine Dienstleistung

traditionelle Web-Anwendung



Web Service





Ein **Web Service** ist eine **Softwareanwendung**, die

1. mit einer **URI** eindeutig identifizierbar ist,
2. über eine **WSDL**-Schnittstellenbeschreibung verfügt,
3. nur über die in ihrer WSDL beschriebenen Methoden zugreifbar ist und
4. über **gängige Internet-Protokolle** unter Benutzung von XML-basierten Nachrichtenformaten wie z.B. **SOAP** zugreifbar ist.

Web Services

- implementieren häufig keine neuen Systeme, sondern sind **Fassade** für bestehende Systeme
- abstrahieren von Programmiersprache und Plattform mit der die Anwendung realisiert ist:
Virtualisierung von Software
- zwei Erscheinungsformen:
 - **RPCs** (synchron)
 - **Messaging** (asynchron)

- Web Services keine revolutionär neue Technologie
- große Ähnlichkeiten mit CORBA

Neu ist jedoch:

- alle bedeutenden IT-Unternehmen auf Standards geeinigt: XML, SOAP und WSDL
- CORBA hingegen nie von Microsoft unterstützt
- statt proprietäre Protokolle (wie IIOP und DCOM) gängige Internet-Protokolle
- nicht nur RPCs, sondern auch Messaging

Layer	Protokoll/Standards
Messaging	HTML
Transport	HTTP, FTP, SMTP
Network	TCP/IP, UDP

Layer	Protokoll/Standards
Content	Content Information
Messaging	SOAP, XML-RPC
Transport	HTTP, FTP, SMTP
Network	TCP/IP, UDP

Web Service Technology Stack

Layer	Protokoll/Standards
Discovery <i>(holt Service-Beschreibung von Prodiver)</i>	UDDI, DISCO, WSIL
Description <i>(beschreibt Services)</i>	WSDL
Messaging	SOAP, XML-RPC
Transport <i>(Applikation-zu-Applikation Kommunikation)</i>	HTTP, FTP, SMPT
Network <i>(Netzwerk Layer)</i>	TCP/IP, UDP



Google™ und Web Services



»With Google Web APIs, your computer can do the searching for you.«

Google als Web Service

- Suche
- Rechtschreibkorrektur
- Zugriff auf Web-Cache

Suche als Web Service

- Suchanfrage als SOAP-Nachricht
- Suchergebnis als SOAP-Nachricht

- Google-Suche kann aus Anwendungsprogramm heraus aufgerufen werden

mögliche Anwendungen

- in periodischen Abständen zu bestimmten Thema nach neuen Webseiten zu suchen:
 - Web Alert
search-for: "XSLT 2.0 Recommendation"
notify new web page: mymail@inf.fu-berlin.de
- automatisch neue Trends im WWW zu identifizieren





amazon.com und Web Services



- ermöglichen Funktionen und Inhalt von Amazon nahtlos in eigene Websites zu integrieren
- konsequente Realisierung einer dienstorientierten Architektur:

If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you

Werner Vogels, CTO von Amazon

- Zugriff über ein **XML-Interface über HTTP** oder ein **SOAP-Interface**
 - Erzeugung von strukturierten Daten (Produktname, Hersteller, Preis etc.) über Produkte von Amazon

- **Infrastuktur Services**
 - Amazon Elastic Compute Cloud
 - Amazon SimpleDB
 - Amazon Simple Storage Service
 - Amazon Simple Queue Service
- **Zahlungs & Abrechnungs-Services**
 - Amazon Flexible Payments Service
 - Amazon DevPay
- **On-Demand Arbeitskraft**
 - Amazon Mechanical Turk
- **Web Suche & Informations-Services**
 - Alexa Web Information Service
 - Alexa Top Sites
 - Alexa Site Thumbnail
- **Amazon & Verbund-Services**
 - Amazon Fulfillment Web Service

- Amazon S3 – Datenspeicherung



Funktionalität

- schreiben, lesen, und löschen von Objekten, die zwischen 1byte-5 GB Daten beinhalten (Anzahl der Objekte, ist nicht limitiert)
- jedes Objekt ist über einen eindeutigen Schlüssel (developer-assigned key) ansprechbar
- Autorsierungsmechanismen (Zugriffsrechte, private & public Objekte)
- Standard-basierte REST & SOAP Interfaces

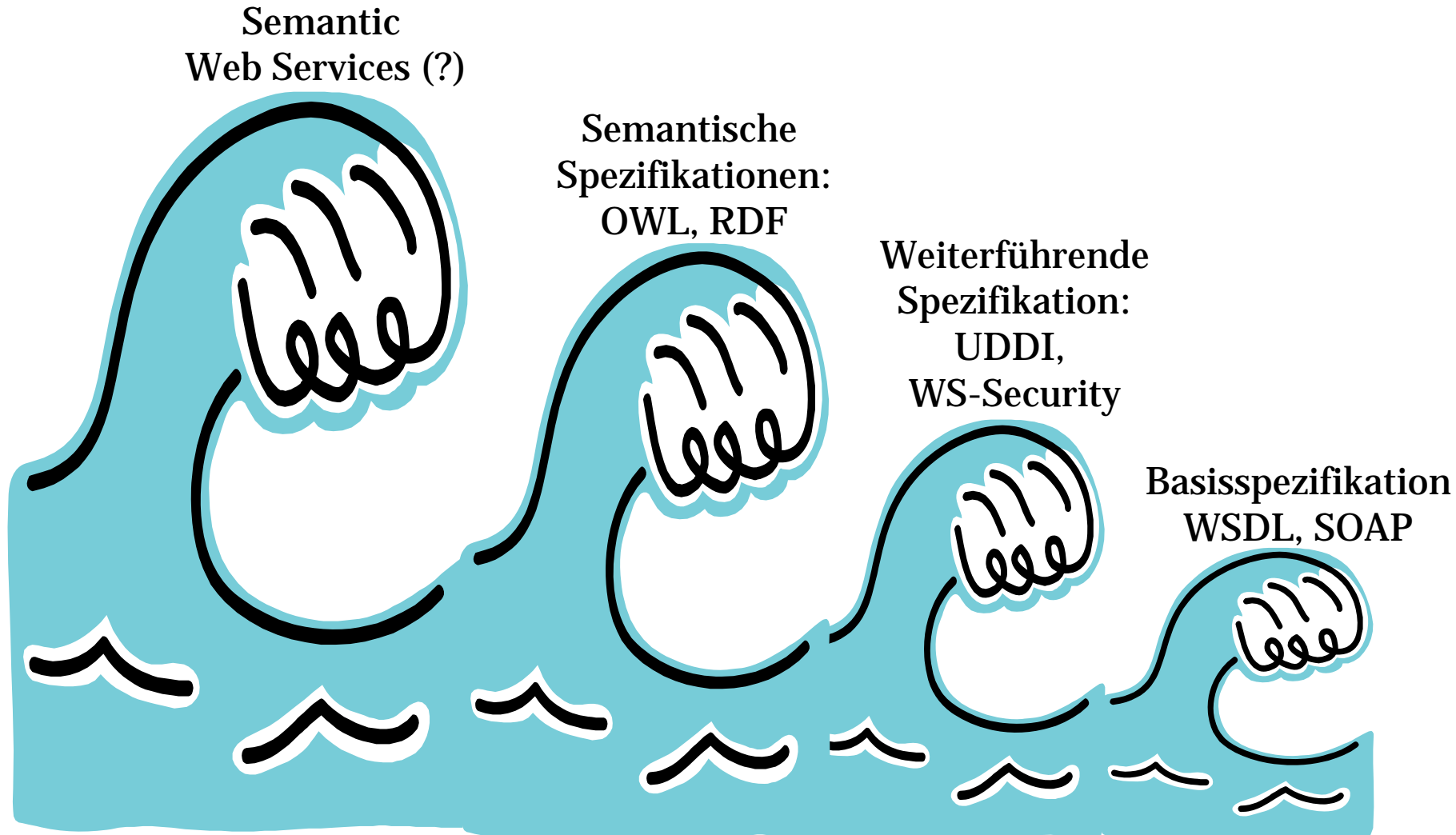
Amazon Flexible Payments Service (Amazon FPS)

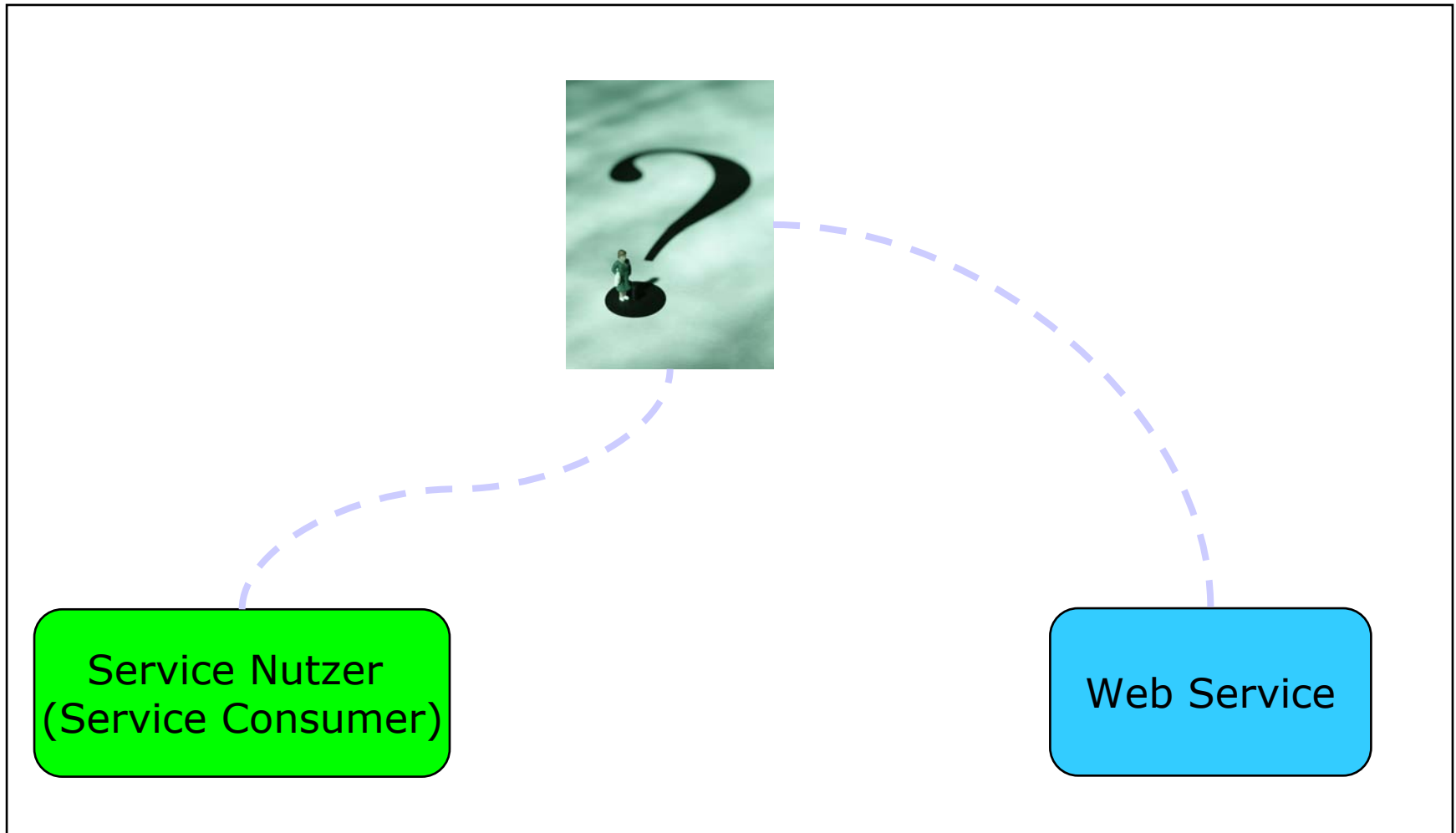
- Set von Web Services APIs
- erlauben Geld-Bewegung zwischen 2 Entitäten (Menschen, Computer)



Funktionalität

- Geld verschicken und empfangen: Kreditkarten, Bankkonten oder Amazon Payments balance transfer
- Durchführung von einmaligen, wiederkehrenden oder multiplen Zahlungen
- Zusammenführen von kleinen Transaktionen in große unter Verwendung von Prepaid und Postpaid-Ressourcen
- Überblick über Kontenniederschriften, Transaktionsgeschichte, und -details
- Nutzen von Amazon FPS sandbox (frei für registrierte AWS Entwickler), um Applikationen ohne echtes Geld zu bilden und testen



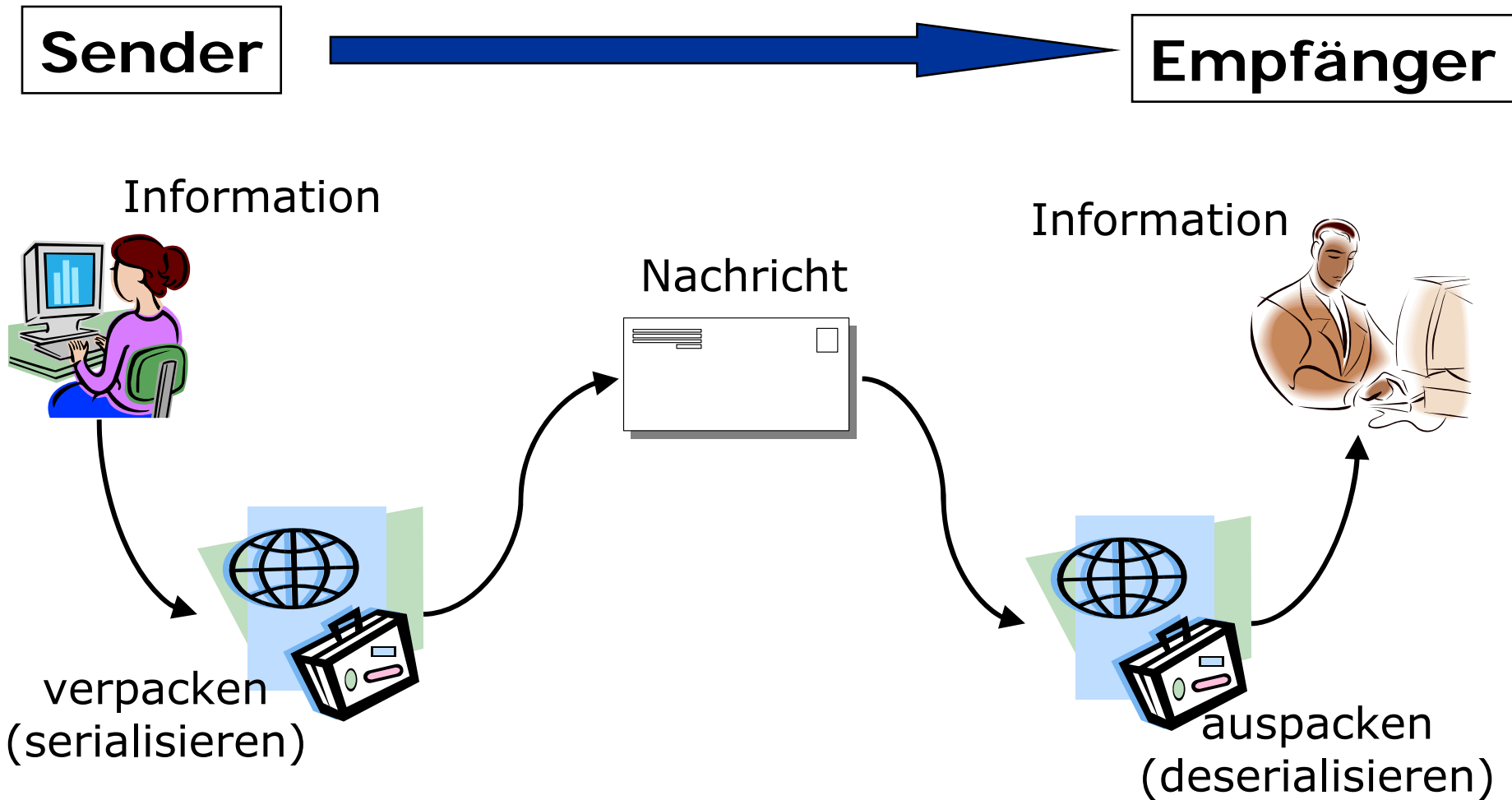




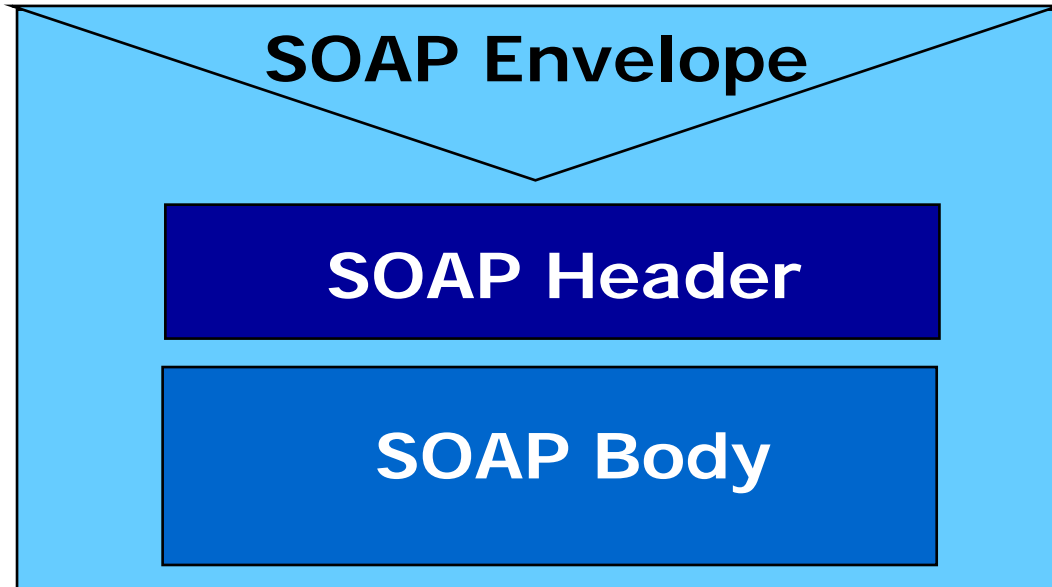
Web Service Basiskomponenten: *SOAP*



Austausch einer SOAP-Nachricht



Aufbau einer SOAP-Nachricht



XML-Deklaration

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
```

```
<!-- SOAP Header -->
```

```
<!-- SOAP Body -->
```

```
</env:Envelope>
```

SOAP Version 1.2

SOAP Version 1.1:
<http://schemas.xmlsoap.org/soap/envelope/>

HTML

```
<html>
  <head>
    Zusatzinformationen
  </head>
  <body>
    Inhalt: Webseite
  </body>
</html>
```

SOAP

```
<Envelope>
  <Header>
    Zusatzinformationen
  </Header>
  <Body>
    Inhalt: XML-Daten
  </Body>
</Envelope>
```

- XML-basierter W3C-Standard
 - SOAP 1.1: W3C Note von 2000
 - SOAP 1.2: W3C Recommendation von 2003
- seit SOAP 1.2: SOAP ≠ Simple Object Access Protokoll


```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="..." xmlns:xsi="...">
  <env:Body>
    <doGoogleSearch xmlns="urn:GoogleSearch">
      <key xsi:type="xsd:string">3289754870548097</key>
      <q xsi:type="xsd:string">Eine Anfrage</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      ...
    </doGoogleSearch>
  </env:Body>
</env:Envelope>
```

- **doGoogleSearch(key, q, start, maxResults,...)**
- hier kein Header
- Beachte: **Web Service-Aufruf kann als URL kodiert werden (REST)**

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="..." xmlns:xsi="...">
  <env:Body>
    <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" ...>
      <return xsi:type="ns1:GoogleSearchResult">
        ...
      </return>
    </ns1:doGoogleSearchResponse>
  </env:Body>
</env:Envelope>
```

- Antwort: **doGoogleSearchResponse**(return(...))
- Datentyp **ns1:GoogleSearchResult** in WSDL-Beschreibung definiert

- kein Protokoll sondern ein **Architekturstil**
- Verwaltet beliebige Menge von Ressourcen
- Minimierung von
 - Netzwerklatenz und
 - Netzwerkverkehr
- Maximierung von
 - Unabhängigkeit und
 - Skalierbarkeit von Komponenten
- **RESTful** → eine Anwendung konform zum REST-Architekturstil

 **amazon.com** – der populärster Anbieter einer REST-Anwendung
(in 2006 ca. 80% der Anfragen über REST)

- REST-Architektur des WWW (Fielding 2000):
jede Web-Ressource soll eindeutig über eine URI identifiziert werden
- Beispiel:
 - online gebuchte Reise = Web-Ressource
 - gebuchte Reise sollte daher auch über eine URI eindeutig identifiziert werden

REST-Prinzipien (I)

1. zustandslose Client/Server-Kommunikation

- Client kann sich nicht darauf verlassen, dass Kontext der vorherigen Interaktion immer noch vorhanden ist
- alle Informationen in der Anfrage

2. identifizierbare Ressourcen

- Ressource - allgemeiner Oberbegriff für konkrete oder abstrakte Dinge, mit denen eine Interaktion möglich ist
- Ressourcen sind über einen standardisierten Mechanismus dauerhaft weltweit eindeutig identifizierbar und adressierbar
- URIs - sichtbarste im Web umgesetzte REST-Prinzip

REST-Prinzipien (II)

3. uniforme Schnittstelle

- nur eine einzige Schnittstellenvereinbarung
- beinhaltet nur die Methoden/Operationen, die prinzipiell von allen Ressourcen unterstützt werden

4. Hypermedia

- „Hypermedia as the Engine of Application State“
- Server überträgt Menge von Daten + die durch den Client initiierbare Zustandsübergänge in Form von eingebetteten Verknüpfungen (*Links*)

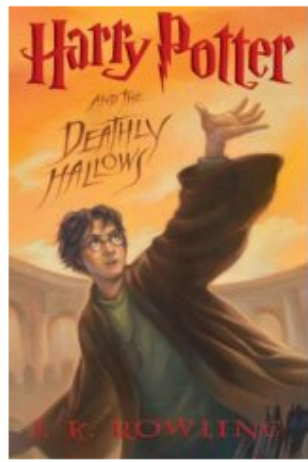
http://ecs.amazonaws.com/onca/xml?Service=AWSECommerceService&AWSAccessKeyId=[ID]&Operation=ItemSearch&SearchIndex=Books&Title=Harry%20Potter

```
- <ItemSearchResponse>
  ...
  - <Items>
    - <Request>
      ...
      - <ItemSearchRequest>
        <SearchIndex>Books</SearchIndex>
        <Title>Harry Potter</Title>
      </ItemSearchRequest>
    </Request>
    <TotalResults>1076</TotalResults>
    <TotalPages>108</TotalPages>
  - <Item>
    <ASIN>0545010225</ASIN>
    - <DetailPageURL>
      http://www.amazon.com/gp/redirect.html%3FASIN=0545010225%26tag=ws%26l
    </DetailPageURL>
    - <ItemAttributes>
      <Author>J. K. Rowling</Author>
      <Creator Role="Illustrator">Mary GrandPré</Creator>
      <Manufacturer>Arthur A. Levine Books</Manufacturer>
      <ProductGroup>Book</ProductGroup>
      <Title>Harry Potter and the Deathly Hallows (Book 7)</Title>
    </ItemAttributes>
  </Item>
  ...
</Items>
</ItemSearchResponse>
```

Ausschnitt der Antwort
des Amazon Servers

Guaranteed Release-Date Delivery—or It's Free Get It July 21—Select Standard Shipping or Sign Up for Amazon Prime

Prime Learn more about Amazon Prime



Harry Potter and the Deathly Hallows (Book 7) (Hardcover)

by J. K. Rowling (Author), Mary GrandPré (Illustrator)

List Price: \$34.99

Price: \$17.99 & eligible for FREE Super Saver Shipping on orders over \$25. Details

You Save: \$17.00 (49%)

Pre-Order Price Guarantee! Order now and if the Amazon.com price decreases between your order time and release date, you'll receive the lowest price. See Details

Availability: This title will be released on July 21, 2007. Pre-order now! Ships from and sold by Amazon.com. Gift-wrap available.

Quantity: 1 Pre-order this item today or Sign in to turn on 1-Click ordering.

- Add to Wish List Add to Shopping List Add to Wedding Registry Add to Baby Registry Tell a friend

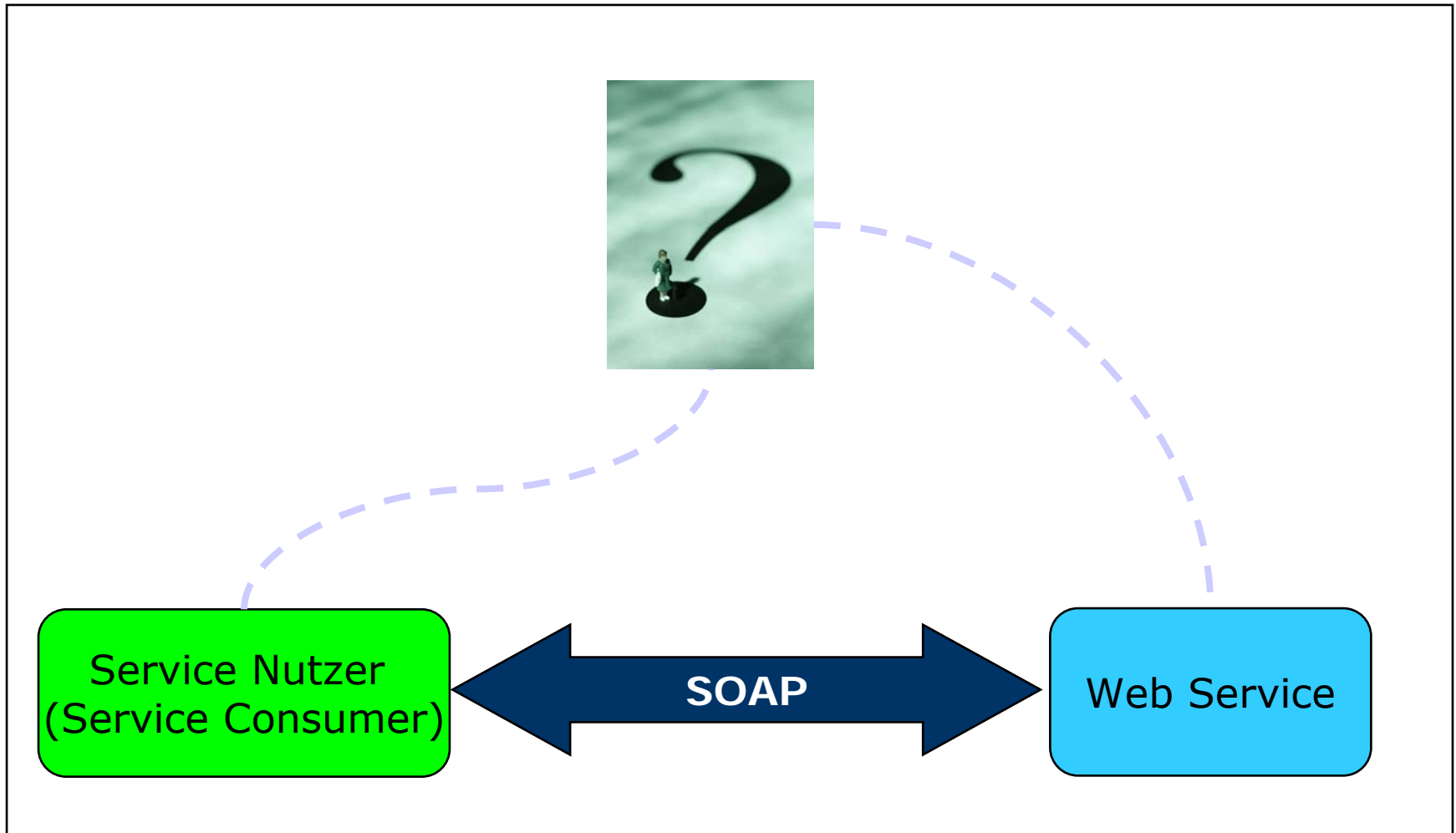
See larger image and other views



Publisher: learn how customers can search inside this book.

Table with 4 columns: Also Available in, List Price, Our Price, Other Offers. Rows include Hardcover (Deluxe), Paperback (Braille), and Library Binding (Library).

- heute meist über HTTP oder HTTPS
- Request-Response-Verhalten von HTTP unterstützt entfernte Prozeduraufrufe (RPCs)
- auch z.B. mit SMTP (Messaging) möglich

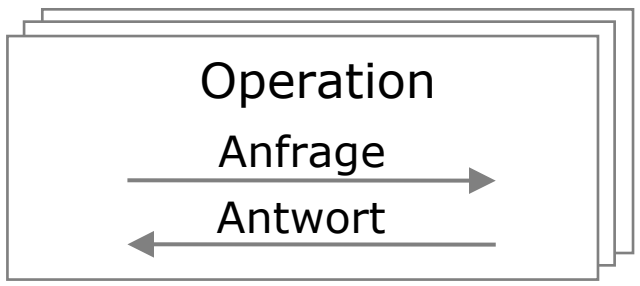




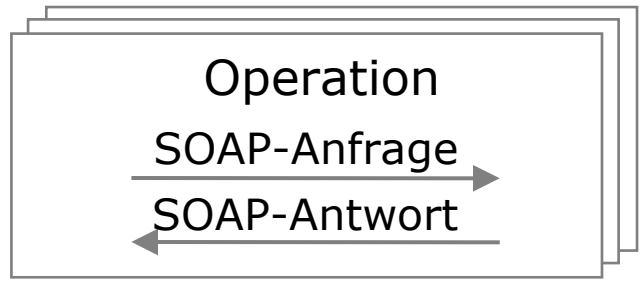
Web Service Basiskomponenten: *WSDL*

Servicebeschreibung

abstrakte Schnittstelle



konkrete Schnittstelle



Web-Adressen (End Points)

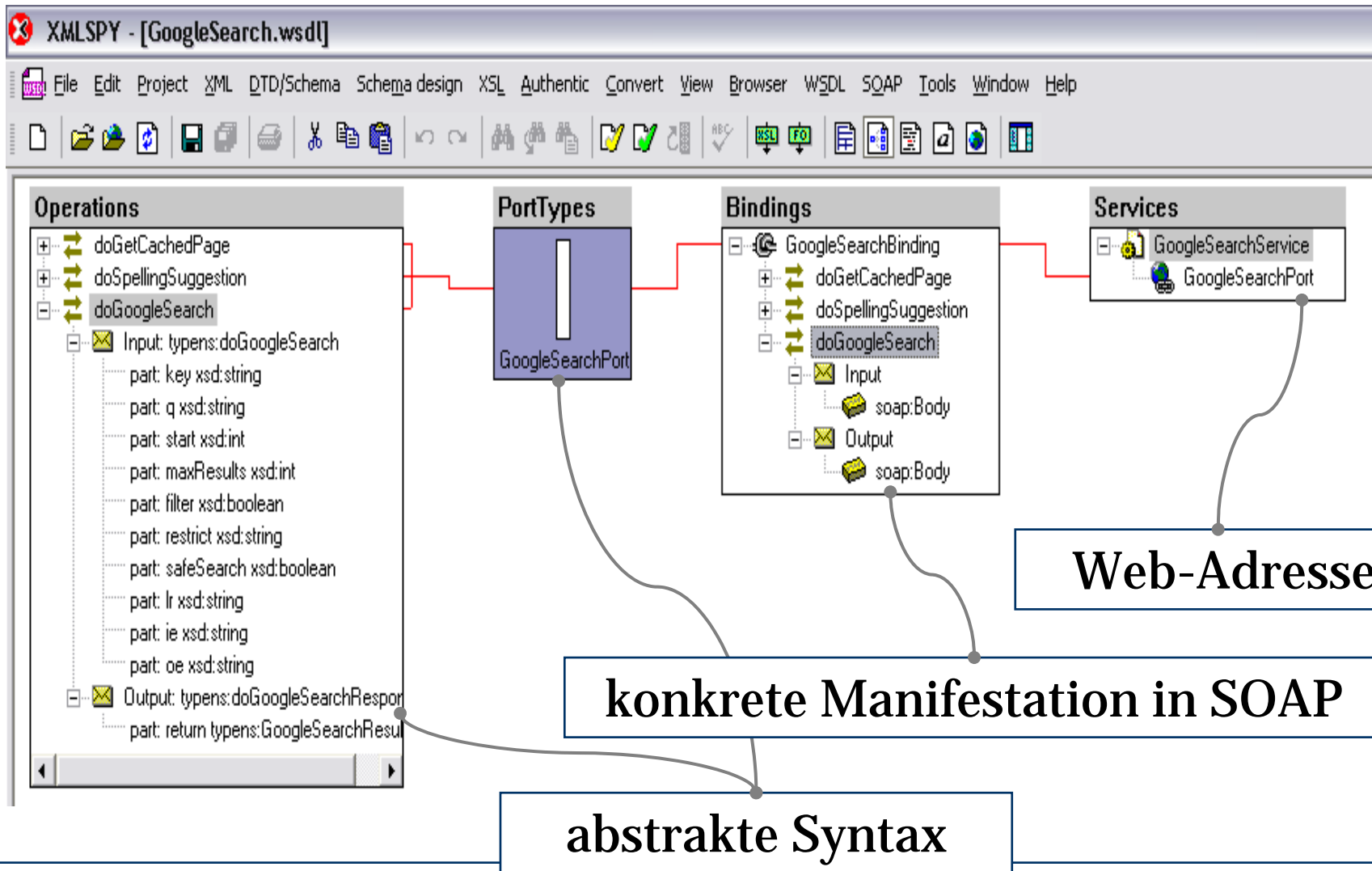
- beschreibt Interface (IDL)
- XML-basierter Standard
- W3C Note von 2001

abstrakte Schnittstelle (port type)

- Schnittstelle unabhängig von Nachrichtenformaten
- Beschreibung mit XML-Schema

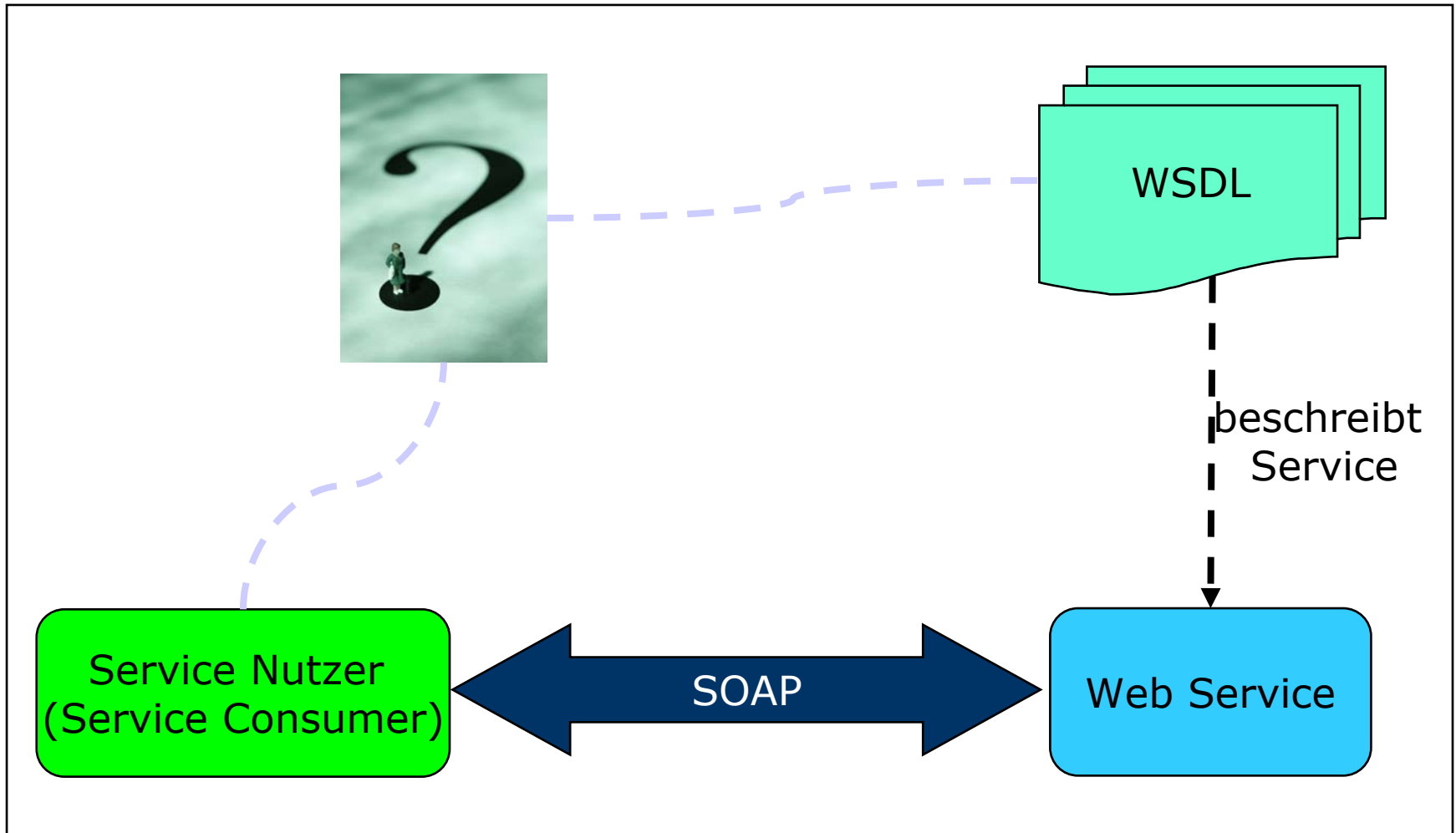
konkrete Schnittstelle (binding)

- Abbildung der abstrakten Schnittstelle auf unterstützte Nachrichtenformate



Eigenschaften von WSDL

- baut auf **XML-Schema** auf
 - ⇒ Syntax einer Schnittstelle kann bis ins kleinste Detail festgelegt werden
- beschreibt **Schnittstelle(n)** eines Web Services und **wo** dieser **abgerufen** werden kann
- **grundlegende Interaktionsmuster** (wie Anfrage-Antwort)
- keine Möglichkeit semantische Eigenschaften zu beschreiben





Web Service Basiskomponenten: *UDDI*

- Universal (Service) Description, Discovery, and Integration
- entwickelt seit Herbst 2000 von Ariba, Microsoft & IBM
- beschreibt einen Verzeichnisdienst für Web Services
- SOAP basierter Dienst

- White Pages
- Yellow Pages
- Green Pages
- Service Type Registration

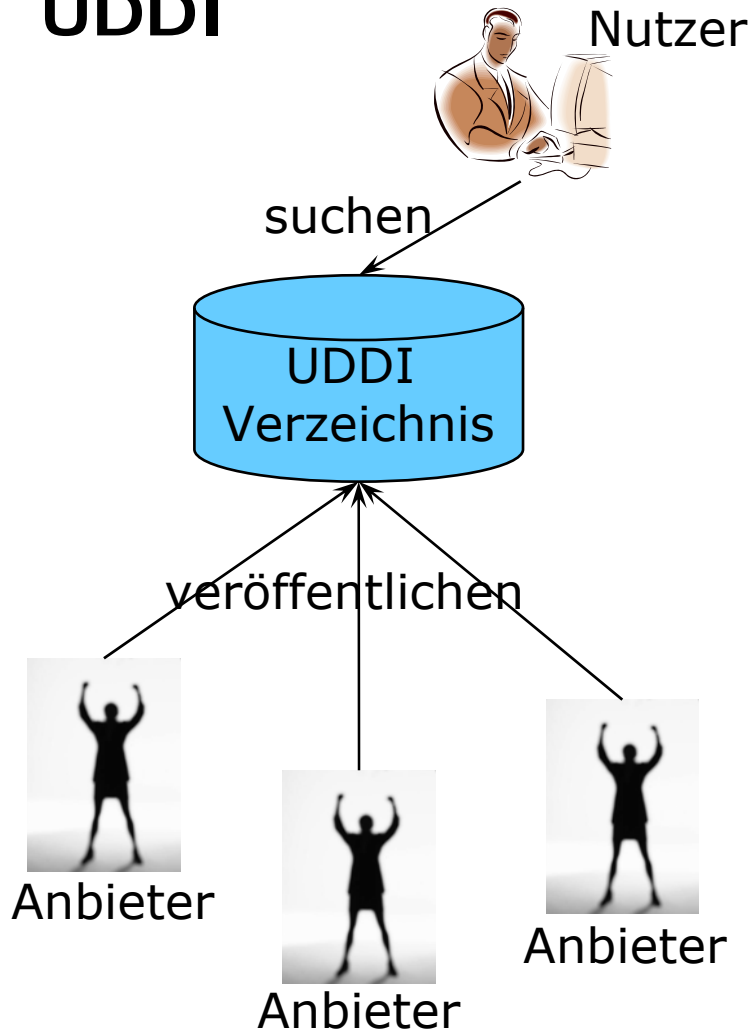


Quelle: <http://www.tecchannel.de/webtechnik/soa/472223/index4.html>

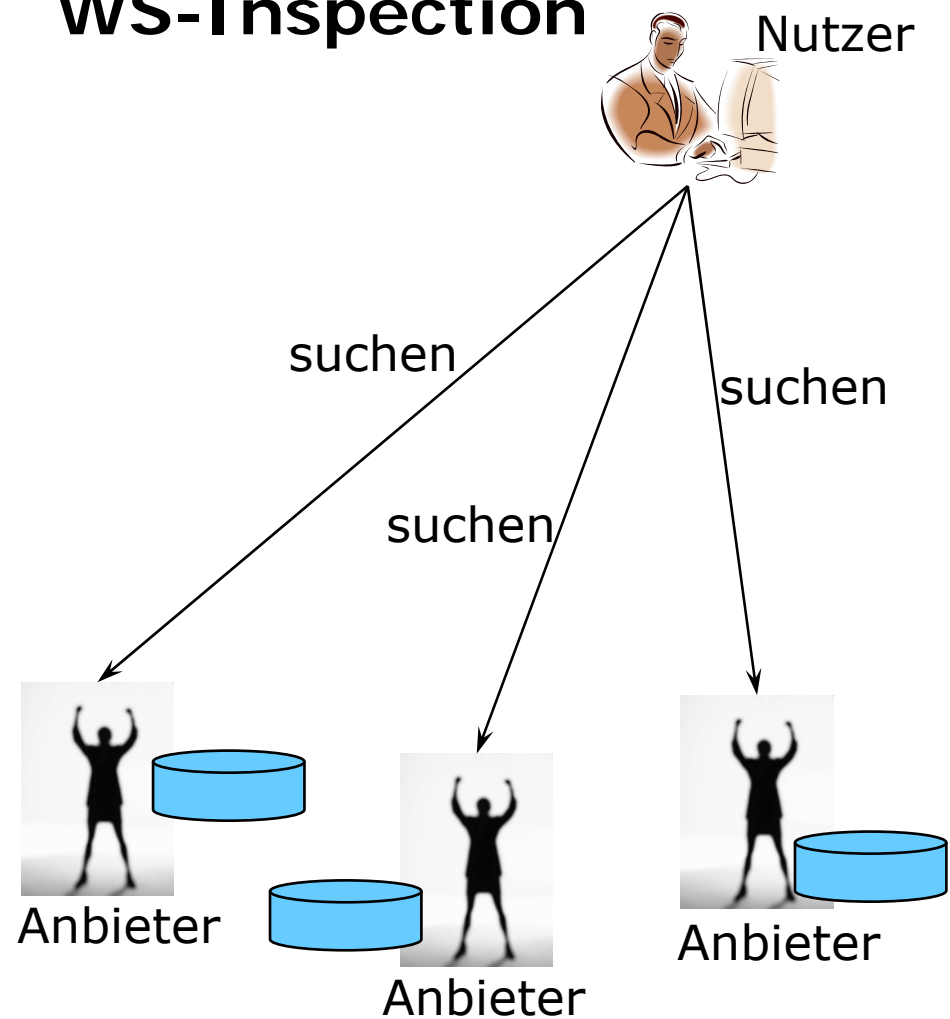
- **UDDI-XML-Schema**
 - Business-Entity
 - Business-Service
 - Binding-Template
 - Technisch Modelle (TModel)
- **UDDI-API**
 - Anwendungsschnittstelle in Form von Web Services

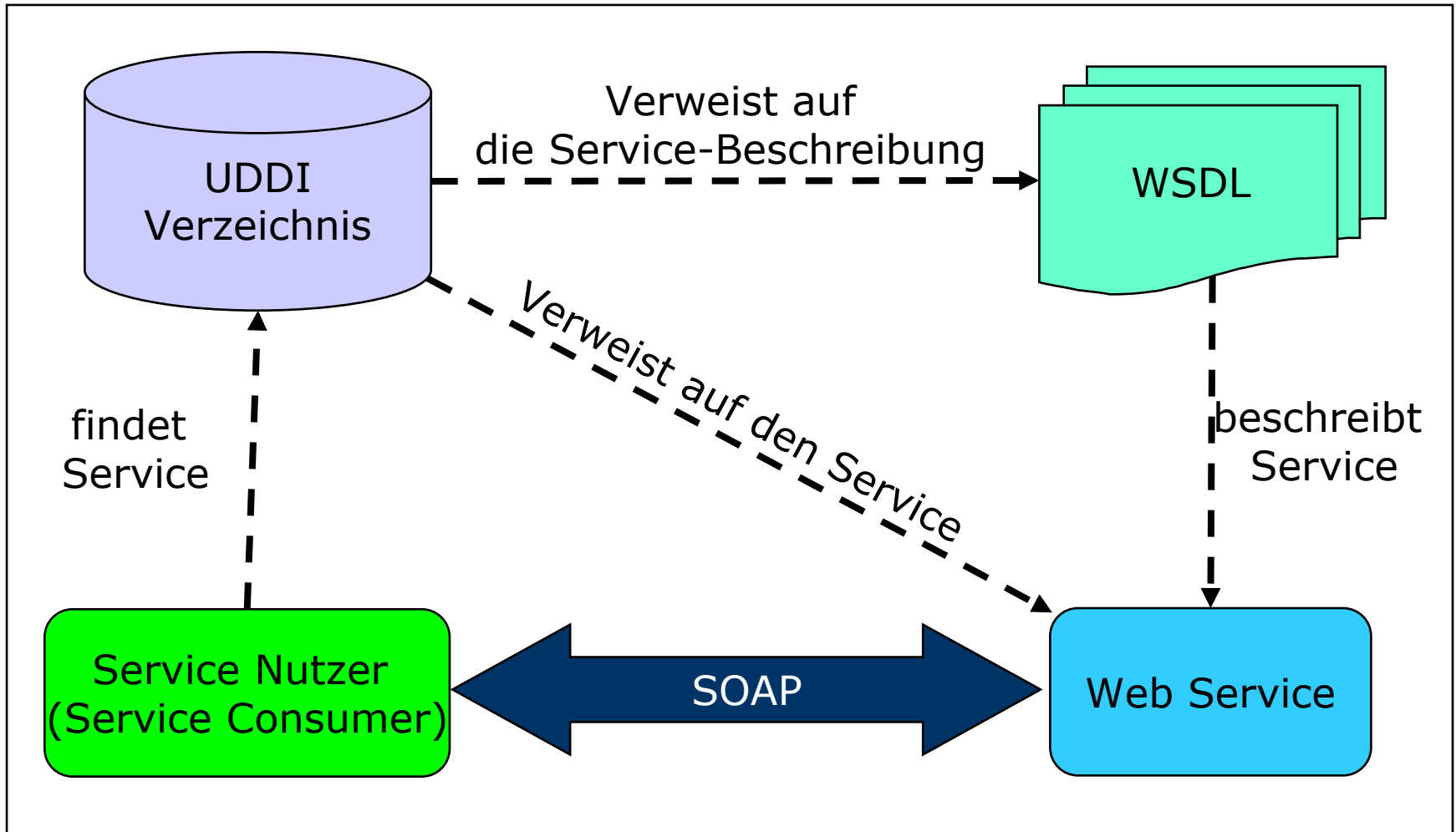
UDDI vs. WS-Inspection

UDDI



WS-Inspection



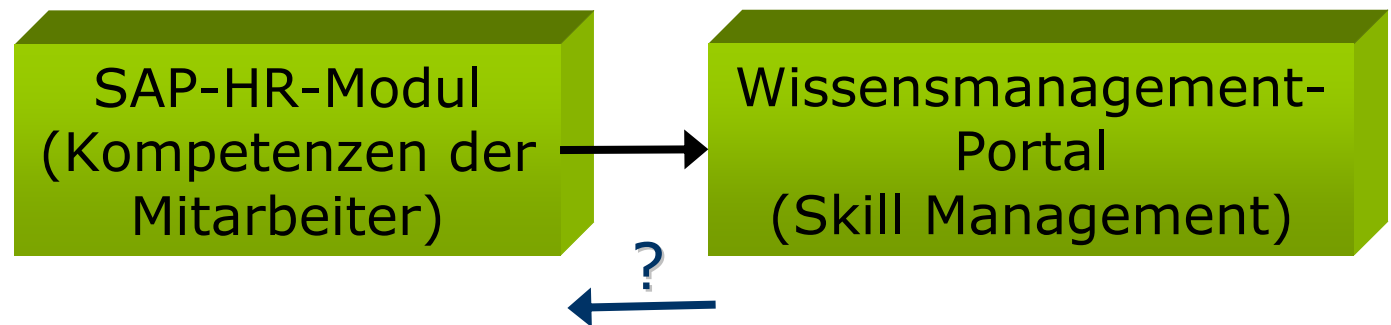




Enterprise Application Integration (EAI)

Enterprise Application Integration

- technologisch: inkompatible IT-Systeme miteinander verbinden
 - inkompatibel kann bedeuten:
 - unterschiedliche Betriebssysteme
 - unterschiedliche Programmiersprachen
 - unterschiedliche Kommunikationsprotokolle
- organisatorisch: Geschäftsprozesse optimieren
- Beispiel:



Zwang zur Systemintegration

unternehmensintern

- **Beispiel Mercedes-Benz-Werk**
 - mehr als 200 EDV-Systeme
 - sehr gut vernetzt (LAN)
 - Systeme also prinzipiell integrierbar
- **Neue Systeme müssen Alt-Systeme integrieren**

unternehmensübergreifend

- **E-Business setzt Zusammenarbeit von heterogenen Systemen voraus:**
 - Unternehmen ↔ Unternehmen
 - Unternehmen ↔ Portal

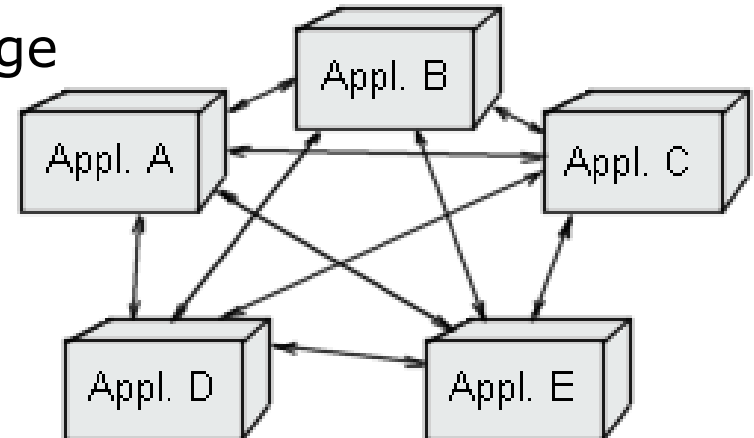
Systemintegration bindet:

- 35% der IT-Personal-Ressourcen eines Unternehmens (Forrester Research, 2002)
- 65% der Arbeitszeit eines Programmierers (Gartner)

Integrationstopologien (I)

Point-to-Point / Peer-to-Peer

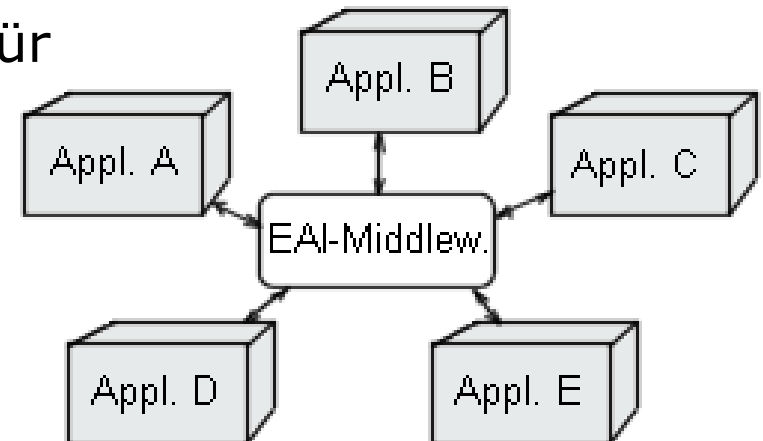
- Spaghettistruktur: Anwendungen werden direkt (1:1) miteinander verbunden
- nur bei wenigen Systemen und wenigen Verbindungen praktikabel
- Vor-/ Nachteile
 - einzelne Systeme nur mit hohem Aufwand austauschbar
 - sehr unflexibel → keine Grundlage für SOA, BPM und Portal
 - hohe Folgekosten dafür aber + geringe Startkosten



Integrationstopologien (II)

Hub & Spoke

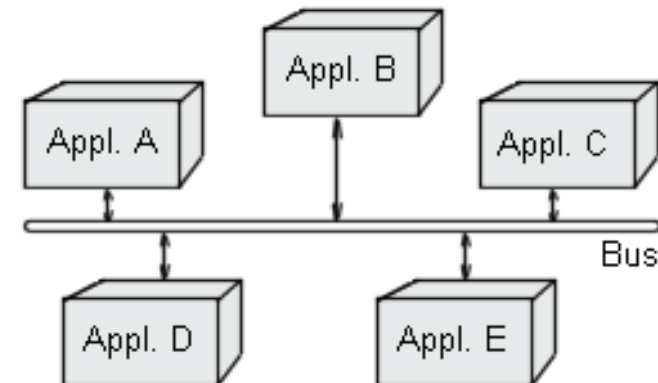
- Nachrichten vom zentralen Hub als Informationsdrehscheibe empfangen, transformiert und weitergeleitet
- geeignet für komplexe Datenverteilungsmechanismen
- zentraler Hub - Performance-Bottleneck, wenn nicht skalierbar
- Vor-/ Nachteile
 - + einzelne Systeme mit geringem Aufwand austauschbar
 - + sehr flexibel → gute Grundlage für SOA, BPM und Portal
 - + geringe Folgekosten aber
 - hohe Startkosten



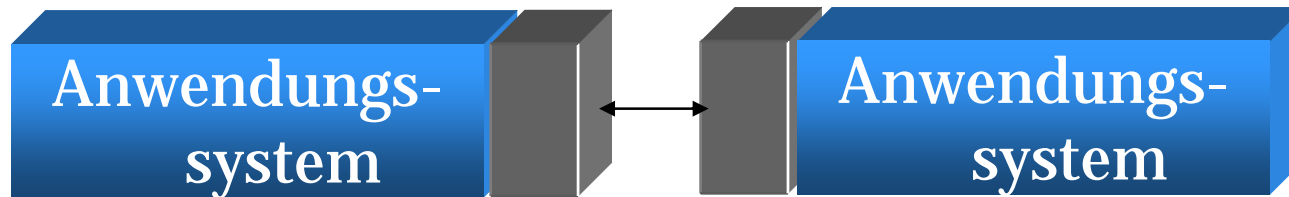
Integrationstopologien (II)

Bus / Pipeline / Publish & Subscribe

- Nachrichten über Bussystem verteilt
- Anbindung an den Bus über verteilte Software-Komponenten, zentrales Repository mit Business Rules
- Vor-/ Nachteile
 - wegen verteilter Architektur aufwändiger
 - einzelne Systeme mit geringem Aufwand austauschbar
 - hohe Startkosten, aber
 - + geringe Folgekosten
 - + sehr flexibel → gute Grundlage für SOA, BPM und Portal

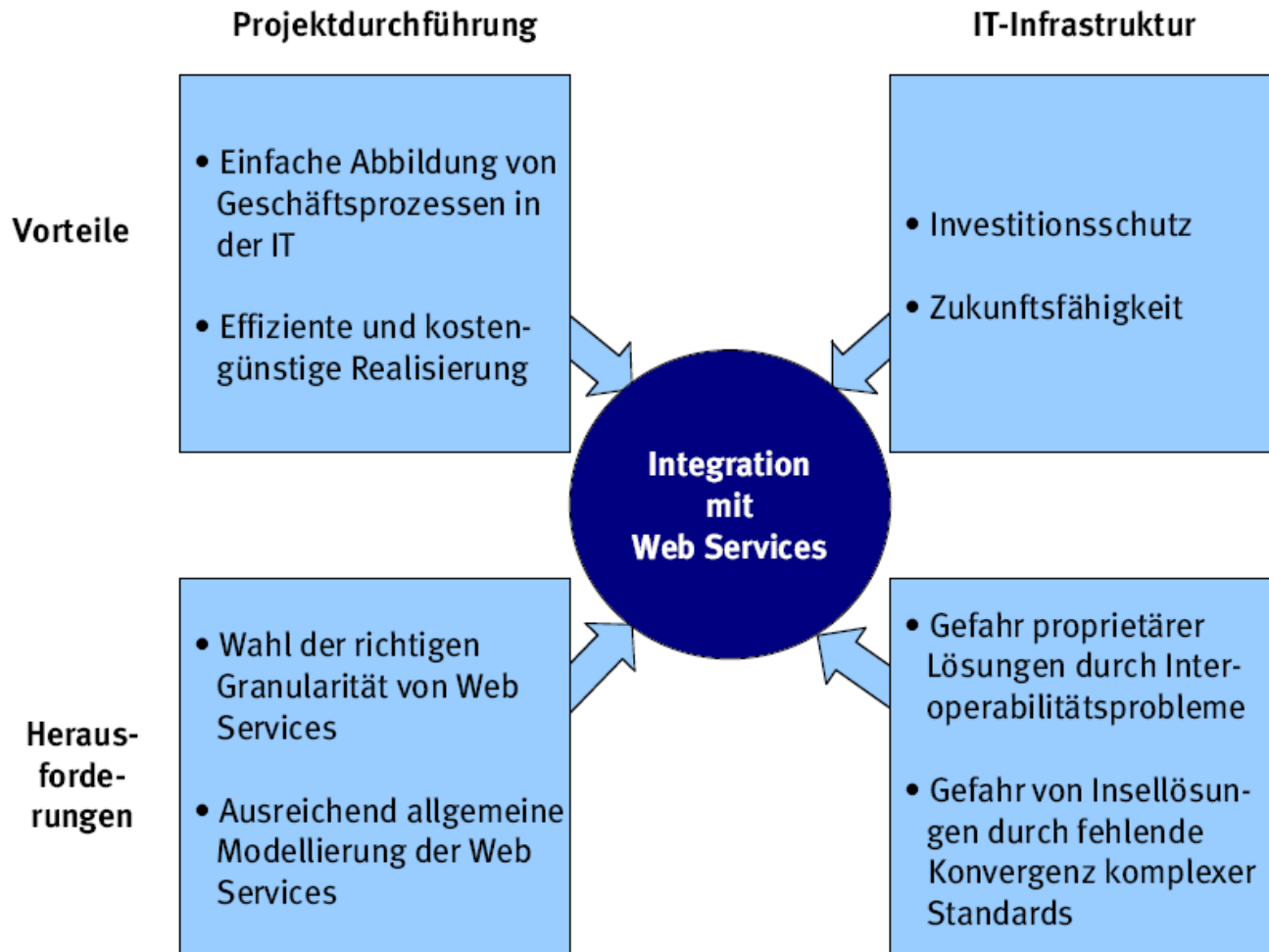


- Anwendungssysteme durch standardisierte Schnittstelle erweitern



- Schnittstelle muss allgemein akzeptiert sein
- bei Web Services ist dies der Fall
 - ✓ SOAP
 - ✓ WSDL
 - ✓ Internet-Protokolle
- bei n Systemen:
statt n^2 Schnittstellen nur n Erweiterungen!

Systemintegration mit Web Services: Vorteile & Herausforderungen



Quelle: Integration mit Web Services -- Konzept, Fallstudien und Bewertung, Berlecon Research, 2003.

Die Nutzung von Web Services erleichtert die Durchführung von Integrationsprojekten, denn:

- Web Services ermöglichen die Wiederverwendbarkeit von Funktionalitäten in unterschiedlichen Zusammenhängen.
- Web Services ermöglichen die Modellierung von Geschäftsprozessen auf einer fachlichen, nichttechnischen Ebene.
- Workflow-Code lässt sich automatisch generieren.
- Legacy-Anwendungen können leicht eingebunden werden.

Quelle: Integration mit Web Services -- Konzept, Fallstudien und Bewertung, Berlecon Research, 2003.

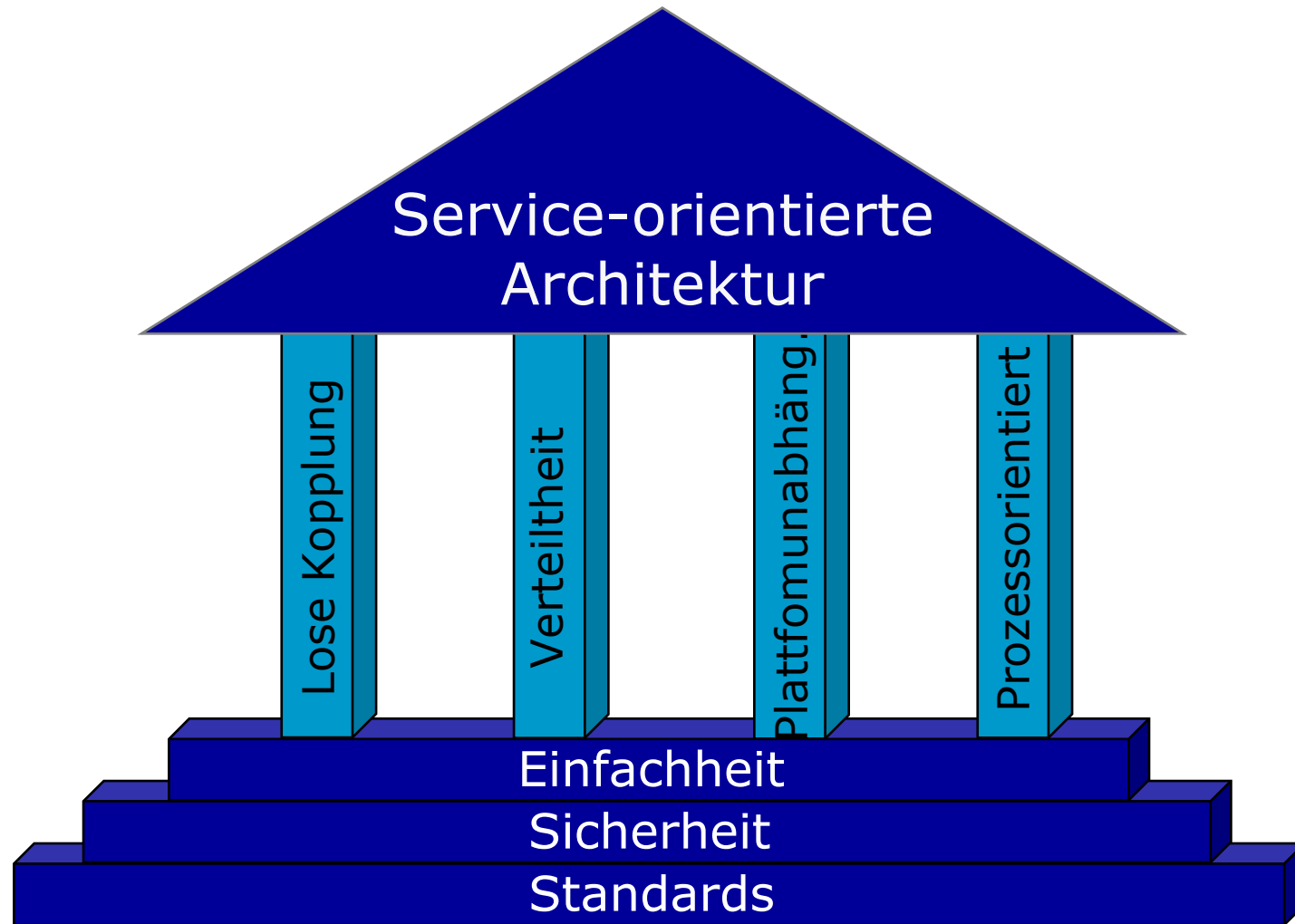


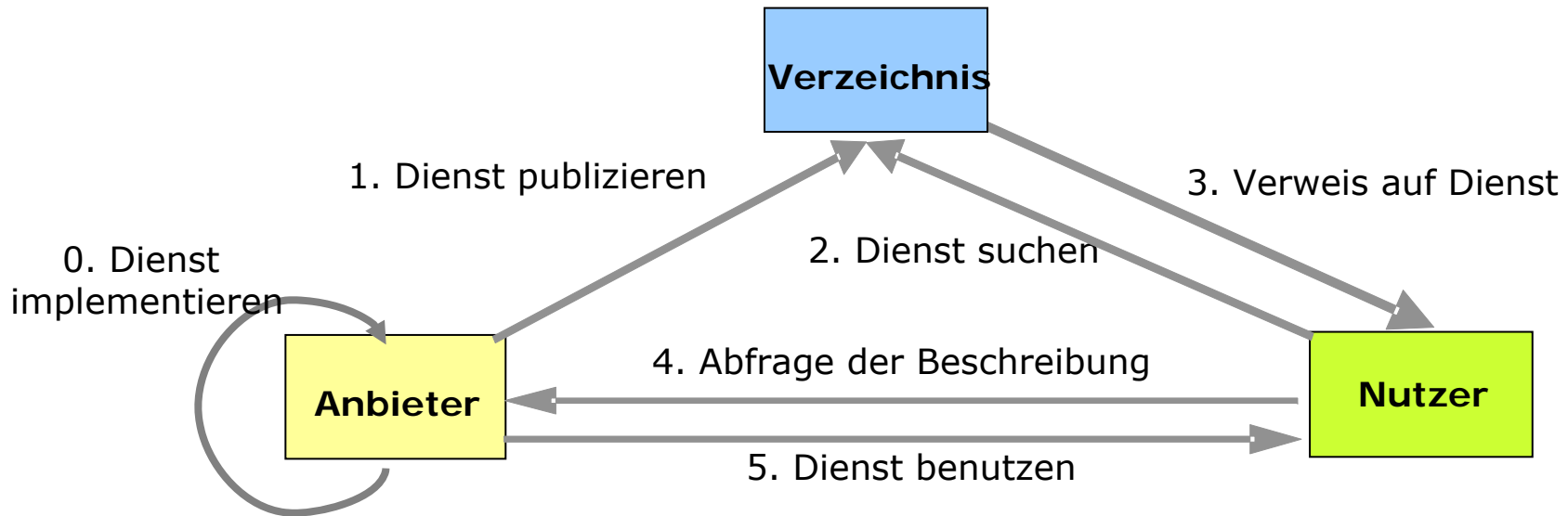
Dienstorientierte Architektur (SOA)

- engl. **service-oriented architecture**, kurz **SOA**
- statt Anwendungen isoliert zu entwickeln, nur um sie später zu integrieren:
- neue Anwendungen von Anfang an auf existierenden Web Services aufbauen
- neue Anwendung wiederum als Web Service anbieten

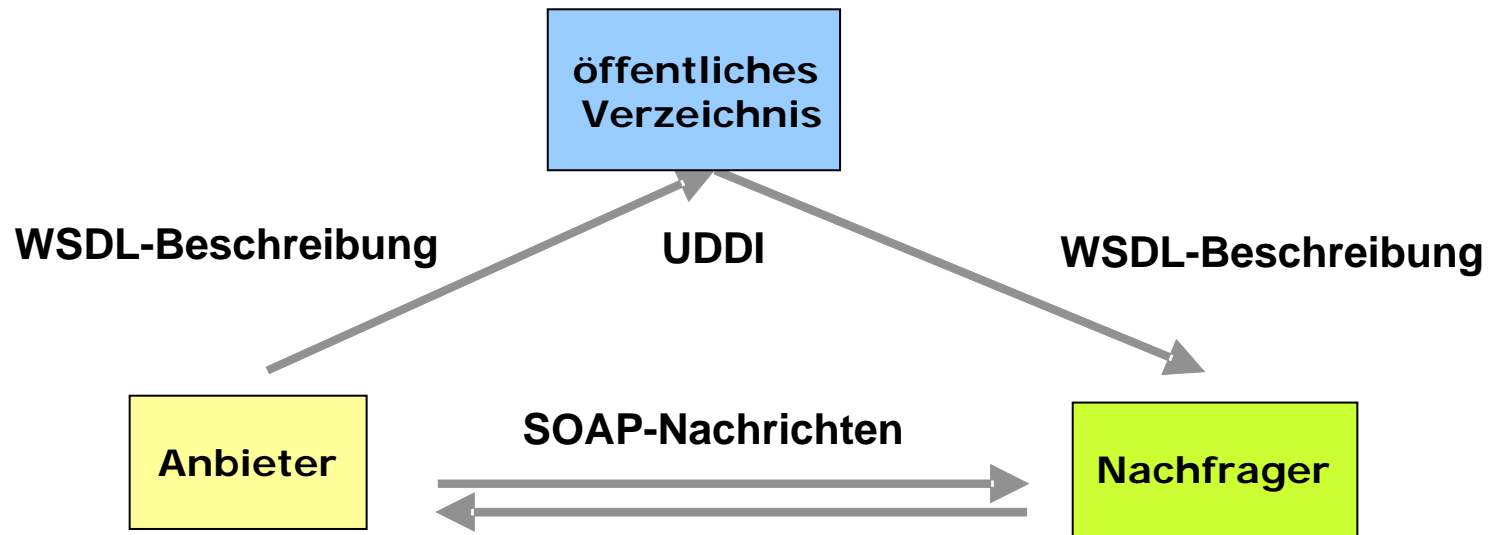
*„... eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachunabhängige Nutzung und Wiederverwendung ermöglicht.“ **

*Quelle: „Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis“, W. Dostal, M. Jeckle, I. Melzer, B. Zengler; Spektrum Akademischer Verlag, 2005

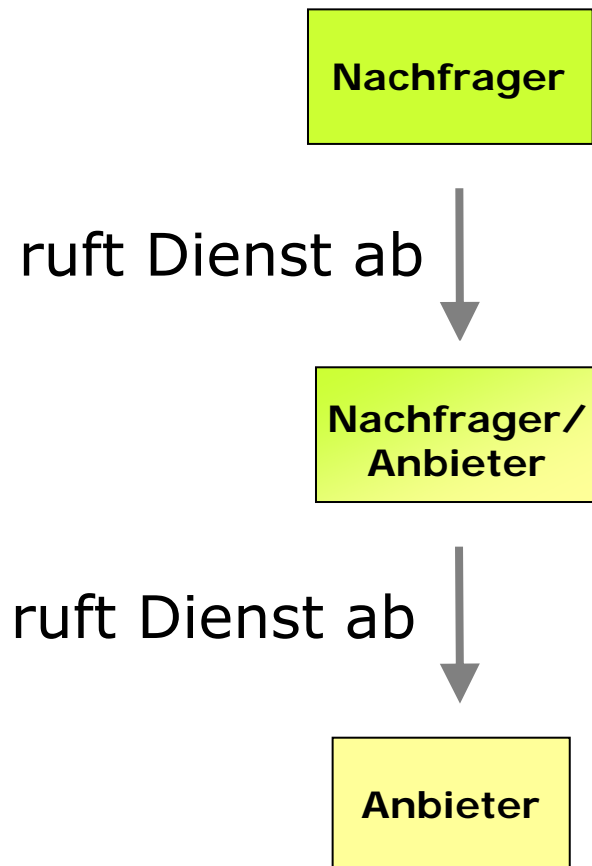




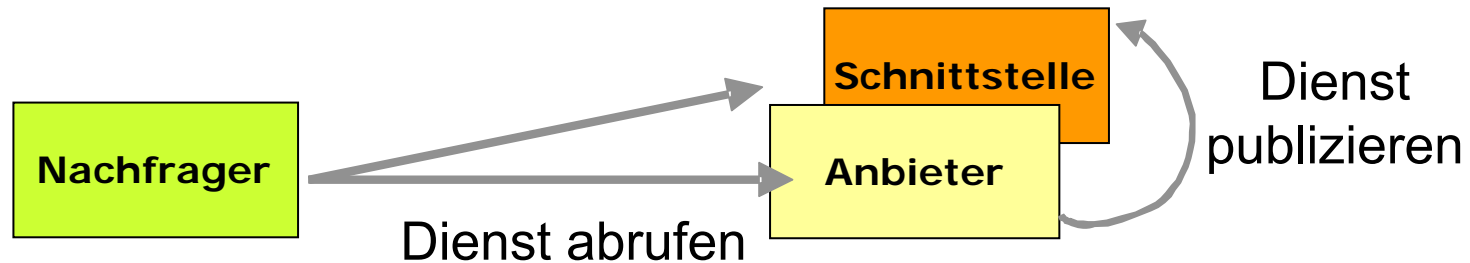
- **publizieren (publish)**: Beschreibung eines Dienstes in einem Verzeichnis (registry) veröffentlichen.
- **suchen (find)**: Beschreibung eines Dienstes suchen, entweder dynamisch oder zur Entwicklungszeit
- **abrufen (bind)**: Beschreibung des Dienstes verwenden, um Dienst abzurufen, entweder dynamisch oder zur Entwicklungszeit



- SOAP und WSDL allgemein akzeptiert
- UDDI: Standard zur Beschreibung von Web-Service-Verzeichnissen
- UDDI umstritten und wenig genutzt



- **Dienst-Anbieter** (service provider) bietet Dienste an.
 - **Dienst-Nachfrager** (service requestor) nutzt Dienste anderer Anbieter.
 - Anbieter von Diensten kann gleichzeitig Dienste anderer nutzen (und Nachfrager) sein.
- ⇒ Diese Begriffe sind relativ.



- Öffentliches Verzeichnis nicht zwingend notwendig:
→ Auch ohne öffentliches Verzeichnis kann Dienst gefunden werden.
- → Schnittstelle kann z.B. auf Webseite des Anbieters veröffentlicht werden.
- öffentliche Verzeichnisse heute wenig genutzt

SOA: Vor- und Nachteile

ohne dass Nutzer des Web Services es bemerkt, kann

- + neue Version freigeschaltet werden
- + bei Ausfall ein Web Service durch einen anderen Web Service mit gleicher Schnittstelle ersetzt werden
- + Lastverteilung durchgeführt werden

allerdings

- Vertrauen nötig, dass Web-Service-Anbieter WSDL-Beschreibung im Sinne des Nutzers realisiert



RPC vs. Messaging

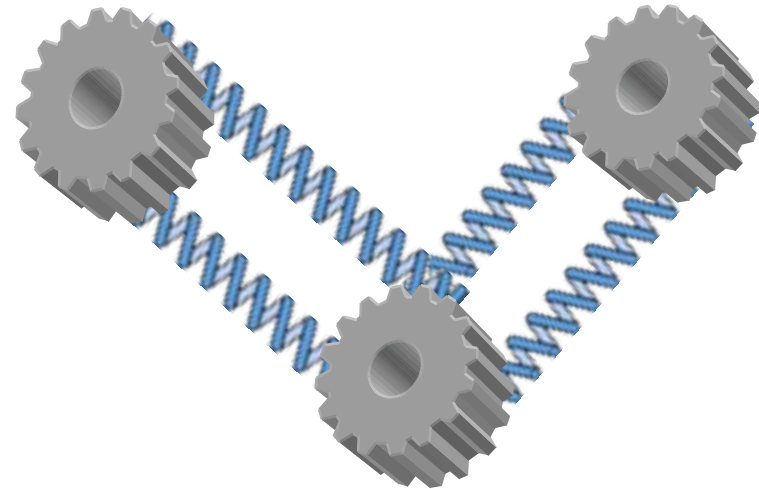
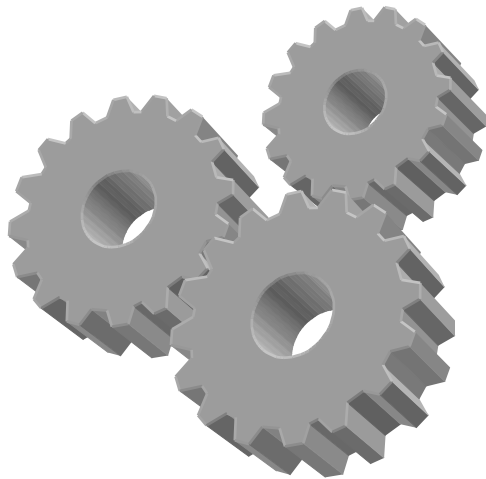
über HTTP

- heute üblich
- Request-Response-Verhalten von HTTP unterstützt **RPCs**
- **verbindungsorientiert**
- Übertragung: Sender und Empfänger müssen präsent sein.
- typischerweise **synchron**

über SMTP

- heute eher selten
- realisiert **Messaging**
- **persistente** Kommunikation
- Übertragung: Weder Sender noch Empfänger muss präsent sein.
- typischerweise **asynchron**
- erlaubt Lastverteilung und Priorisierung

⇒ weitreichende Designentscheidung!



enge Kopplung

- **verbindungsorientierte, synchrone Kommunikation**
- z.B. SOAP/HTTP

lose Kopplung

- **gepufferte, asynchrone Kommunikation**
- z.B. SOAP/SMTP
- robuster, aber auch komplexer zu entwerfen

- **Nachrichten konform:**
 - zu einer vordefinierten Beschreibung des Funktionsaufrufes
 - zu der zu erwartenden Rückantwort
- **Arten der Kommunikation:**
 - Anfrage (Request)
 - Antwort (Response)
 - Fehlerfall (Fault)

asynchroner Nachrichtenaustausch

- auch nach Timeout Antwort noch möglich
- Was tun mit der Antwort?
- häufig muss Absender informiert werden, dass Antwort nicht verarbeitet werden konnte
- Was tun, wenn diese Warnung nicht rechtzeitig beim Absender ankommt?
- Absender hat vielleicht im falschen Glauben, dass seine Antwort verarbeitet wurde, weitergearbeitet.
- Dieser Vorgang muss dann evtl. rückgängig gemacht werden.

- Anwendungen interagieren durch Austausch von Nachrichten miteinander:
- Kommunikation in Anwendung sichtbar: senden und empfangen
- verschiedene Formen des Messaging:
 1. Kommunikationsstruktur
 2. Interaktionsmuster
 3. flüchtig vs. persistent
 4. synchron vs. asynchron
 5. Quality of Service

1. Kommunikationsstruktur

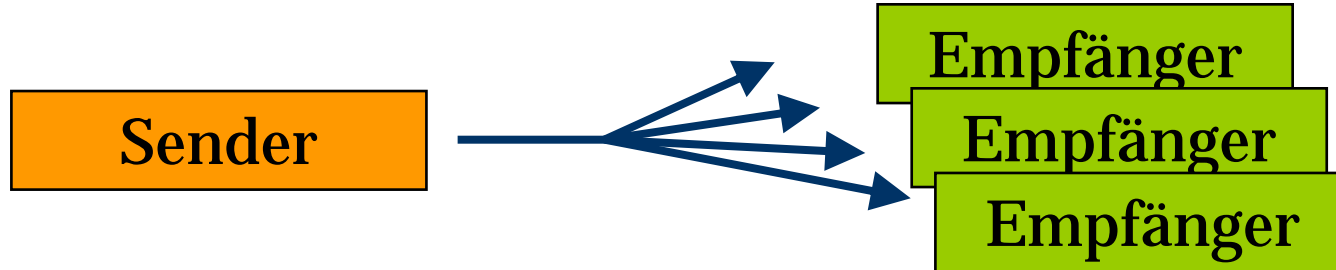
- Anzahl und Organisation der Kommunikationspartner
- wichtigste Kommunikationsstrukturen:
 - One-to-One-Kommunikation
 - One-to-Many-Kommunikation

1. Kommunikationsstruktur: One-to-One-Kommunikation



- Sender sendet Nachricht an bestimmten Empfänger.
- Beispiel: Kunde sendet Bestellung per E-Mail an Firma

1. Kommunikationsstruktur: One-to-Many-Kommunikation



- Sender sendet identische Kopie gleichzeitig an mehrere Empfänger
 - Sender (publisher) veröffentlicht Nachricht zu bestimmten Thema (topic), zu dem sich die Empfänger (subscriber) angemeldet haben.
- ⇒ auch **publish-subscribe** oder **topic-based messaging** genannt
- Beispiel: Mailing-Liste

2. Interaktionsmuster

Client  Server

Einweg (one way,
fire and forget)

Client  Server

Anfrage-Antwort
(request-response)

Client  Server

Benachrichtigung
(notification)

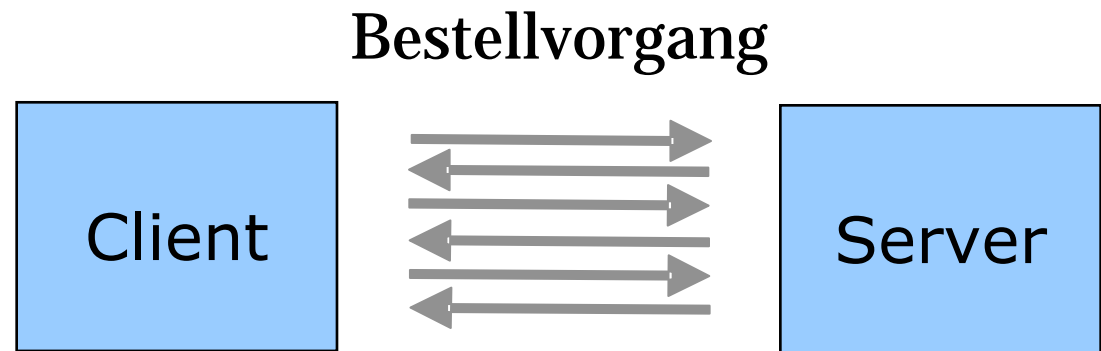
Client  Server

**Benachrichtigung-
Antwort** (notification-
response)

Beachte: „Antwort“ bezieht sich auf jeweilige Anwendung,
nicht auf das Netzwerk.

2. Interaktionsmuster: Komplexe Interaktionsmuster

- Bestellanfrage mit spätestem Liefertermin
- ← Angebot mit zugesichertem Liefertermin
- Bestellung
- ← Bestätigung des Eingangs der Bestellung
- Bestätigung der Lieferung
- ← Rechnung



3. Flüchtig vs. Persistent



flüchtige Kommunikation

- Sender und Empfänger kommunizieren direkt ohne Puffer miteinander.
- Sender und Empfänger müssen während der gesamten Übertragung präsent sein
- engl. transient
- Beispiele: HTTP, UDP

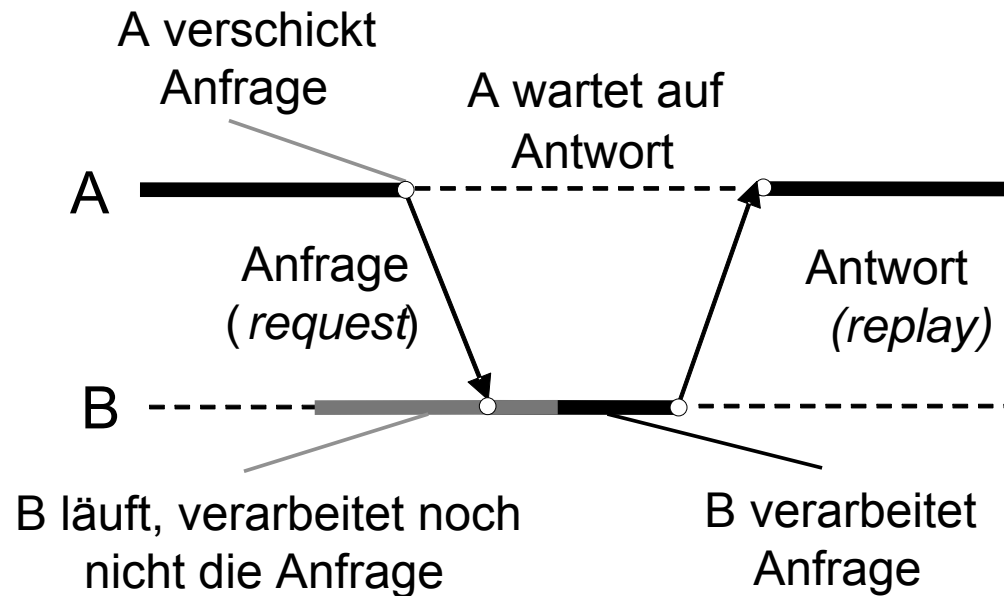


persistente Kommunikation

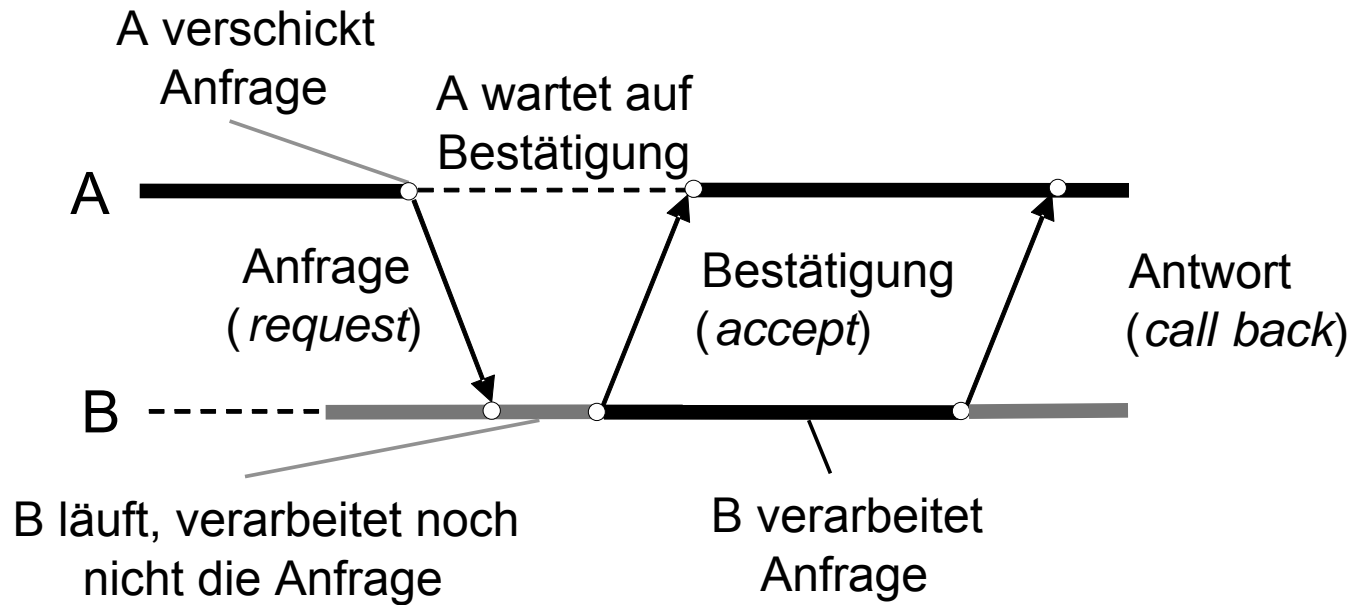
- Nachricht solange gespeichert, bis sie tatsächlich zugestellt wurde.
- Weder Sender noch Empfänger müssen während Übertragung präsent sein.
- Beispiele: E-Mail, MQSeries (IBM), JMS

4. Synchron vs. Asynchron

- **Asynchron**
 - Senden und Empfangen zeitlich versetzt
 - ohne Blockieren des Prozesses (ohne Warten auf die Antwort)
- **Synchron**
 - Senden/Empfangen synchronisieren → warten (blockieren) bis die Kommunikation abgeschlossen ist.
- bei **persistenter Kommunikation** → Messaging normalerweise asynchron
- bei **flüchtiger Kommunikation** → Messaging kann sowohl synchron als auch asynchron sein



- **flüchtige, synchrone** Kommunikation
- auch **antwortorientiert** (response-based) genannt
- implementiert synchrone RPCs
- jedoch \neq RPC: Kommunikation für Anwendung sichtbar



- **flüchtige, asynchrone** Kommunikation
- auch **bestätigungsorientiert** (delivery-based) genannt
- implementiert asynchrone RPCs
- jedoch \neq RPC: Kommunikation für Anwendung sichtbar

RPC oder Messaging?

RPC

⇒ **eng gekoppelte, starre Systeme**

- + einfach, abstrahiert von Kommunikation
- nur Eins-zu-Eins-Kommunikation
- Client und Server müssen präsent sein
- skaliert weniger gut

Messaging

⇒ **lose gekoppelte, robuste Systeme**

- abstrahiert nicht von Kommunikation
- + erlaubt auch One-to-Many-Kommunikation
- + Weder Sender noch Empfänger müssen präsent sein
- + erlaubt Priorisierung und Lastverteilung
- + skaliert sehr gut

heutige Vorlesung

- ☑ Was sind Web Services?
- ☑ Basistechnologien (SOAP, WSDL, UDDI)
- ☑ Enterprise Application Integration
- ☑ Dienstorientierte Architektur (SOA)
- ☑ RPC vs. Messaging

nächste Vorlesung

- SOAP im Detail